SYSTEM RELIABILITY THROUGH FAULT TREE ANALYSIS

TOM BINGHAM
STATISTICIAN
ASSOCIATE TECHNICAL FELLOW
THE BEOING COMPANY
N 8TH & PARK AVE N
RENTON, WA 98055

SUMMARY

Reliability Analysis has become a collection of graphical and mathematical tools designed to model and predict risks associated with deliverable products or services relative to their overall purpose. Such tools vary in sophistication and application, but they generally share a common approach, that being to determine cause and effects between simple events and complex outcomes. This paper will focus on one of those tools and the methods supporting it called ***Fault Tree Analysis.***

INTRODUCTION

To perform a fault tree analysis, one determines some outcome or time based mission that must succeed with very high reliability that is the probability of failure must be very low. Such missions are often so critical, that one would really prefer to say that failure is impossible rather than highly improbable, but such perfectly fail safe thinking can never be assured with 100% confidence. Therefore we content ourselves with building models, that themselves are conservative in structure, and can show that the risk of failure is so low that one cannot imagine it occurring.

MODELING RELIABILITY

Fault Tree Analysis is directly related with classical probability theory. In its essence, it is a fairly simple concept. One builds simple boolean relationships among various events. At the core, relationships fall mostly into the following two camps, namely:

1. Event A ***and*** event B will cause event C.
2. Event D ***or*** event E will cause event F.

An experienced analyst will probably take issue with the above simplicity, and for good reason, but most of the additional complexity of a fault tree still has the above two classifications in mind – so for the time being we will stick with that.

Looking at life from an AND or OR world view, a tree structure becomes an obvious way to display a succession of failure paths. If C & F can cause yet some other event to fail, one could imagine a hierarchy of failure paths that lead us from very basic events to more complex ones. After tracing through all such relationships, one could build a tree (albeit it is upside down), where the ends of is branches represent simple components with known failure rates and its main trunk represents the top event which we demand a nearly impossibly low probability of failure.

To take this from the abstract to the concrete, suppose we are considering entrusting the safety of our three year old daughter to a tricycle. We certainly want to minimize the risk of her getting hurt, and to protect her from the risk of the tricycle failing in a way that could cause her to crash. To use a fault tree model, we must first identify all components on the tricycle that if failed, could contribute to a crash.

Table 1 show a list of such components, along with ficticous failure rates associated with each. We will assume that the parts fail according to an exponential distribution using the tabled rates, and that all components are mutually independent.

Table 1
Tricycle Components Affecting Operation Reliability

Component	Failure Rate	Description
XFRAME	1.11E-03	FRAME FAILS STRUCTRUALLY
XLGRIP	1.11E-02	LEFT GRIP DELAMINATES
XRGRIP	1.11E-02	RIGHT GRIP DELAMINATES
XSEAT1	1.11E-02	SEAT MATERIAL BREAKS
XCLAMP	1.11E-01	CLAMP MECHANISM FAILS
XCBOLT	1.11E-01	STEERING CLAMP BOLT FAILS
XCNUT	1.11E-02	STEERING CLAMP NUT HAS INSUFFICIENT CRIMP
XFDLM	1.11E-02	FRONT WHEEL DELAMINATION
XPBLT1	1.11E-01	SEAT TO POST BOLT 1 FAILS
XPNUT1	1.11E-01	SEAT TO POST NUT 1 HAS INSUFFICENT CRIMP
XPBLT2	1.11E-01	SEAT TO POST BOLT 2 FAILS
XPNUT2	1.11E-01	SEAT TO POST NUT 2 HAS INSUFFICIENT CRIMP
XLDLM	1.11E-03	LEFT REAR WHEEL DELAMINATION
XRDLM	1.11E-03	RIGHT REAR WHEEL DELAMINATION
XHUB1	1.11E-02	LEFT REAR HUB HAS INSUFFICIENT CRIMP
XRAXL	1.11E-01	REAR AXLE FAILS
XHUB2	1.11E-02	RIGHT REAR HUB HAS INSUFFICIENT CRIMP
XFBLT1	1.11E-02	POST TO FRAME BOLT 1 FAILS
XFNUT1	1.11E-02	POST TO FRAME NUT 1 HAS INSUFFICIENT CRIMP
XFBLT2	1.11E-02	POST TO FRAME BOLT 2 FAILES
XFNUT2	1.11E-02	POST TO FRAME NUT 2 HAS INSUFFICIENT CRIMP
XFAXL	1.11E-04	FRONT AXLE BREAKS
XFLNUT	1.11E-02	FRONT LEFT NUT HAS INSSUFICIENT CRIMP
XFRNUT	1.11E-02	FRONT RIGHT NUT HAS INSUFFICIENT CRIMP
XFDELM	1.11E-05	FRONT WHEEL DELAMINTES

Note: All failure rates are ficticous and are included here only to demonstrate fault tree analysis techniques.

As a second step, we will create cause and effect conditions that relate failure of these components with definable events. These events will in turn be combined with other cause and effect relationships that ultimately find their way to the top event, that being a crash of the tricycle. Table 2 shows such cause and effect relationships. Boolean operations are shown as either AND or OR . These logical conditions are generally called “gates”. The gate name is defined on the leftmost column. It may be used repeatedly in any of the child columns. This then defines relationship among gates and components that define the propagation of potential failure.

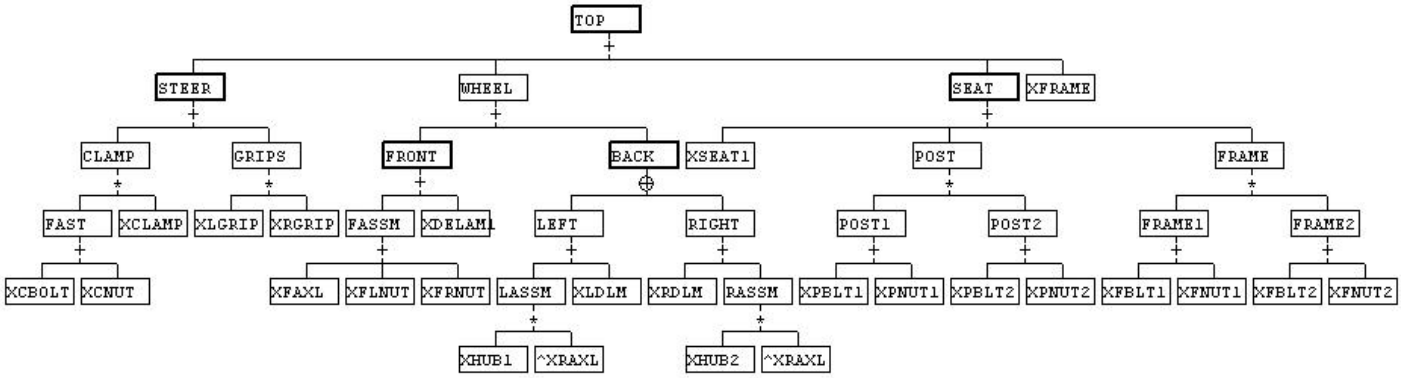
Table 2
Cause & Effect Relationships Defining the Fault Tree

Parent	Description	Gate	Child 1	Child 2	Child 3	Child 4
TOP	TRICYCLE CRASHES IN SERVICE	OR	STEER	WHEEL	SEAT	XFRAME
WHEEL	WHEEL FAILURE	OR	FRONT	BACK		
BACK	EXACTLY ONE REAR WHEEL FAILS CAUSING REAR FRAME TO HIT THE GROUND AND TWIST	XOR	LEFT	RIGHT		
GRIPS	GRIPS DELAMINATE	AND	XLGRIP	XRGRIP		
SEAT	SEAT FAILS TO SUPPORT OPERATOR	OR	XSEAT1	POST	FRAME	
STEER	STEERING FAILS OR CONTROL IS LOST BY OPERATOR	OR	CLAMP	GRIPS		
CLAMP	CLAMPING ASSEMBLY ALLOWS STEERING MECHANISM TO LOOSEN	OR	FAST	XCLAMP		
FAST	STEERING FASTENERS FAIL	OR	XCBOLT	XCNUT		
FRONT	FRONT WHEEL FAILS SO AS TO LOSE SUPPORT OF FORWARD FRAME	OR	FASSM	XFDLM		
POST	SEAT TO POST ATTACHMENT FAILS	AND	POST1	POST2		
POST1	SEAT TO POST FASTENER ASSEMBLY 1 FAILS	OR	XPBLT1	XPNUT1		
POST2	SEAT TO POST FASTENER ASSEMBLY 2 FAILS	OR	XPBLT2	XPNUT2		
LEFT	WHEEL FAILS CAUSING LEFT REAR FRAME TO HIT THE GROUND	OR	LASSM	XLDLM		
LASSM	LEFT REAR WHEEL ASSEMBLY FAILS	AND	XHUB1	XRAXL		
RIGHT	WHEEL FAILS CAUSING THE RIGHT REAR FRAME TO HIT GROUND	OR	XRDLM	RASSM		
RASSM	RIGHT REAR WHEEL ASSEMBLY FAILS	AND	XHUB2	XRAXL		
FRAME	POST TO FRAME ATTACHMENT FAILS	AND	FRAME1	FRAME2		
FRAME1	POST TO FRAME ASSEMBLY 1 FAILS	OR	XFBLT1	XFNUT1		
FRAME2	POST TO FRAME ASSEMBLY 2 FAILS	OR	XFBLT2	XFNUT2		
FASSM	FRONT AXLE FAILS	OR	XFAXL	XFLNUT	XFRNUT	

Note that by convention, components are indicated in the event name by the name with "X". In addition, "Z" is often used when the designation is preliminary.

By defining the possible cause and effect relationships, reliability engineers can model failure paths from product components to mission failure. Table 2 does a good job documenting all Boolean relationships, but fails to yield a visual sense of the failure paths. This can be accomplished (generally through software) by using the relationships in table 2 to create an inverted tree structure with the mission even at the top, and the components at the bottom. Each event connecting the bottom with the top are displayed through connecting branches. Figures 1illustrates the resulting fault tree generated by the relationships in table 2.

Figure 1
Cause & Effect Relationships Illustrated as a Fault Tree



conditions are generally called ¨gates¨ Other symbols are used for more complex gates, such as the exclusive OR gate shown under node" BACK"..

The display in figure 1 greatly improves the visibility of failure paths, although it runs the risk of getting very busy as the tree increases in size and complexity. Various conventions exist to improve the clarity, all of which require a series of coordinated pages to make the display useful. The above "stick figure" graphical approach is also typically replaced with larger boxes to allow the inclusion of the event descriptions. At this point we generally call each of these boxes "nodes", since that term fits the tree structure well. The node called "TOP" on this tree corresponds to the event of the tricycle crashing while in use.

In addition to the obvious benefit to clarity, it is useful to break the tree into segments or *branches*. This is especially useful if the branch selected is self contained, meaning that no nodes in that branch are found anywhere else in the tree. Such *"independent*" branches are also probabilistically independent from the rest of the tree, and it becomes meaningful to calculate the reliability using the top of that branch as a top event of its own. This branch typically has a useful interpretation to the reliability of the system. We will also see that there is a very practical benefit that we will glean after we discuss the next topic concerning failure paths.

FAILURE PATHS AND MIN CUT SETS

Our next step will be to determine paths in the fault tree that lead to mission failure. Conceptually this is fairly straight forward. One must identify which set of components that, if fail simultaneously, drives failure to the top of the tree. One method to accomplish this is to flag every combination of all bottom events as ***failed***, and check each such combination (or set) for mission failure. Those sets that resulted in the TOP event failing are called "cuts". A collection of different cuts is called a "cut set". We might be interested in the set of all possible cuts. This would determine all possible failure scenarios, however there are a few complexities. First, it does not make sense, and in fact causes some trouble, to include a cut in a cutest if it has a subset which is also a cut. In other words, if [X1, X2] is a valid cut, then so is [X1,X2,X3]. But including X3 brings nothing to the table since it is not necessary to fail the mission. Furthermore, keeping such supersets takes extra computer time and memory (which we will find is more damaging than one might imagine), but it also complicates the probability calculations. Consequently, we will seek to eliminate such supersets. Algorithms to eliminate supersets get a bit involved, and we will not entertain them here, but suffice it to say that when an exhaustive set of cuts have been found, and all their supersets eliminated, the resulting cutest is called the "***min cut set***". Table 3 shows the min cut set for our tricycle application.

Table3
Min Cut Set for Tricycle Application

	Expanded List	Mission	
No	TOP	1.018E-01	Rnk
1	[XFAXL]	1.109E-04	7
2	[XFDLM]	1.103E-02	1
3	[XFLNUT]	1.103E-02	1
4	[XFRAME]	1.109E-03	5
5	[XFRNUT]	1.103E-02	1
6	[XLDLM]	1.109E-03	5
7	[XRDLM]	1.109E-03	5
8	[XSEAT1]	1.103E-02	1
9	[XCLAMP,XCBOLT]	1.103E-02	2
10	[XCLAMP,XCNUT]	1.159E-03	3
11	[XFBLT1,XFBLT2]	1.218E-04	6
12	[XFBLT1,XFNUT2]	1.218E-04	6
13	[XFNUT1,XFBLT2]	1.218E-04	6
14	[XFNUT1,XFNUT2]	1.218E-04	6
15	[XHUB1,XRAXL]	1.159E-03	3
16	[XLGRIP,XRGRIP]	1.218E-04	6
17	[XPBLT1,XPBLT2]	1.103E-02	2
18	[XPBLT1,XPNUT2]	1.103E-02	2
19	[XPNUT1,XPBLT2]	1.103E-02	2
20	[XPNUT1,XPNUT2]	1.103E-02	2
21	[XRAXL,XHUB2]	1.159E-03	4

Twenty-one cuts comprise the min cut set. Each has its probability of failure shown under the "mission" column, along with the importance of the cut determined through a ranking of the mission failure probabilities.

Using the min cut set shown in table 3, we have a means to calculate the mission unreliability, which is the probability that a crash occurs. The procedure to do this rests on very basic probability theory. The min cut set shown in table 3 can be thought of as a simplification of the fault tree. The probability that at least one such cut fails (mission unreliability) can be represented as a logical OR for all 21 cuts. The probability that any given cut occurs is the event that all of its components fail. Therefore the min cut set itself is an equivalent fault tree (probabilistically speaking) to the original, consisting of a single OR gate and as many AND gates as cuts in the set.

CALCULATING THE RELIABILITY

The unreliability (1- reliability) is the probability of the top event failing. Calculating this probability is another straightforward task that also can get to be unwieldy for even medium sized trees. But in theory, we can do the following:

Given any two events, say A & B, the probability of A or B occurring is:

(1) P[A or B] = P[A] + P[B] – P[A,B]

This formula applies to any two events, independent or not. A and B can be basic product components, but they can also be cuts. This formula can be used directly to calculate the probability of the entire set of cuts occurring which is the mission failure probability. Furthermore, this approach can be generalized to any number of events (or cuts), although the calculations get cumbersome. For example:

(2) P[A or B or C] = P[A] + P[B] + P[C] – P[A,B]– P[A,C]– P[B,C]+ P[A,B,C]

And also:

(3) P[A or B or C or D] = P[A] + P[B] + P[C] + P[C]
− P[A,B]– P[A,C]– P[A,D]– P[B,C]– P[B,D]– P[C,D]
+ P[A,B,C]– P[A,B,D]– P[A,C,D]– P[B.C,D]
– P[A,B,C,D]

Even with only four events one can see the complexity growing. The number of sets to evaluate approximately doubles for each new event added. For the 21 cuts in our tricycle example, the number of set combinations that must be processed is 2,097,151. Although modern computers could easily handle the scope of that, it does not take too many more before even the most powerful computers will fail to manage the size of the calculation. If our problem grows to 30 cuts, the number of sets to evaluate grows to 1,073,741,823 which will require 500 times more computer resources than when we had only 21 cuts.

The good news is that most of these sets contribute very little to the final probability, albeit they occur in vast numbers. The second source of good news is that some cancellation occurs due to the oscillating sign in the expression. This lends itself to approximate formulas that work well, provided one does not include supersets in the cutest itself, that is we use a min cut set. One such approximation is to simply add the first order terms in the probability expression. If the failure rates are small, say 0.00001 or less, the higher order terms are orders of magnitude below the first order terms, and even thought they may exist in vast numbers, their cumulative effect is still dominated by the first order terms. This greatly reduces the scope of the calculations.

Another approximation which is a little better makes use of the following relationship:

(4) P[A or B] ≈ P[A] + P[B] – P[A]*P[B].

This becomes an equality rather than an approximation if A & B are independent, which is not necessarily the case since A & B are cuts that may share common components. But since we are using a min cut set, A & B must contain some components that are unshared, thus driving down the size of the resulting probability, particularly for components with low failure rates.

Now note that:

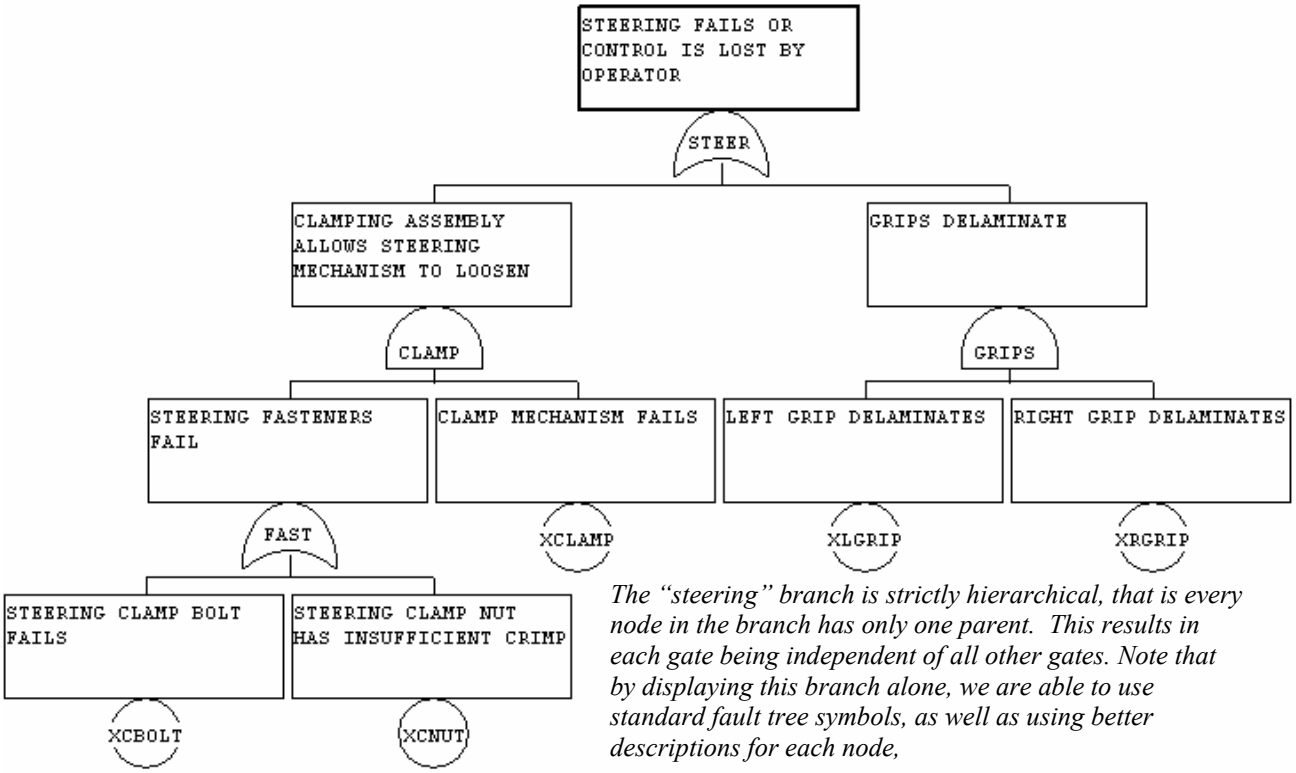
(5) P[A or B or C]=P[(A or B) or C]
≈ P[A or B] + P[C] – P[A or B]*P[C].
= P[A or B] *(1-P[C]) + P[C].

Lending itself to an iterative calculation that takes about the same effort as just adding the first order effects. The result is shown at the top of table 3 as the mission TOP failure probability.

FAULT TREE COMPLEXITIES

One might question why the formula in equation 4 is not exact if applied hierarchically throughout the tree. For example, if we looked at Steering branch (see figure 2) we would find a fairly simple structure. This branch controls the steering. Each gate and component in this branch is found in this branch only. This exclusiveness results in a strictly hierarchical tree, and the entire substructure within the branch is probabilistically independent from any event outside the branch. This means that equation (4) could be applied at each node, using only the cuts at that node, and getting an exact probability for that gate. Furthermore, we can apply that probability to this gate and treat it as if it were another component. As we continue to move up the tree, we need not carry the baggage of the cuts at that point, but rather imagine the tree to be pruned at that branch, and treat the node name as a single component.

Figure 2
Steering Branch: Simple Hierarchy



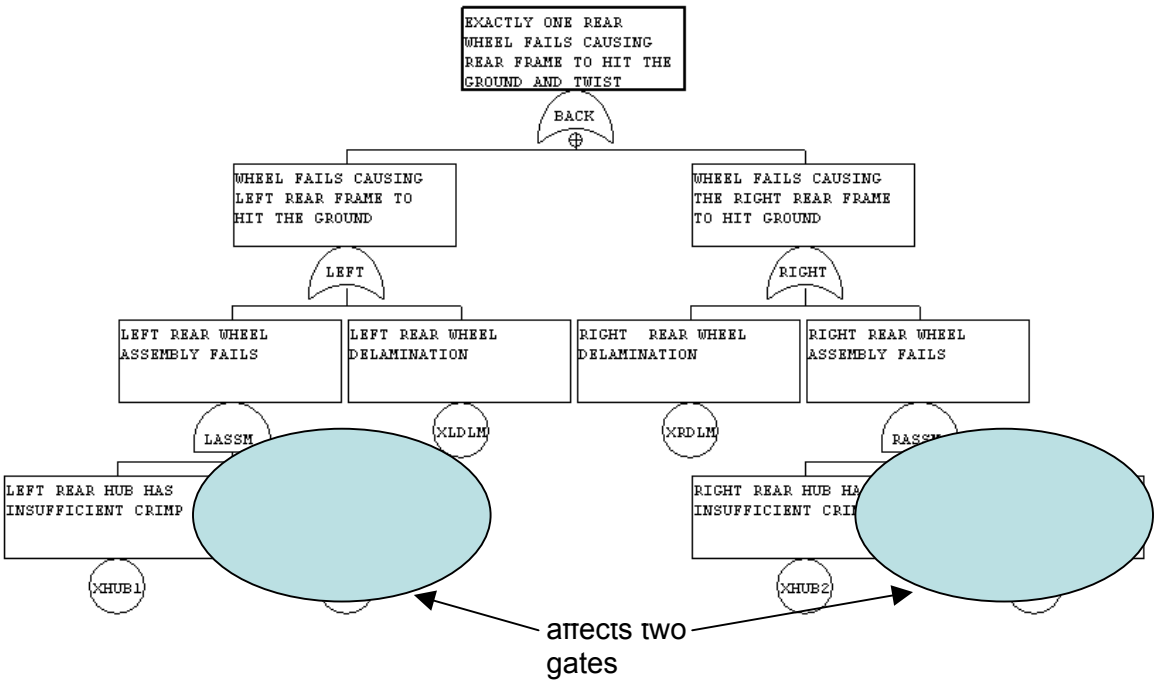
The "steering" branch is strictly hierarchical, that is every node in the branch has only one parent. This results in each gate being independent of all other gates. Note that by displaying this branch alone, we are able to use standard fault tree symbols, as well as using better descriptions for each node,

This approach not only drastically reduces the complexity of the probability calculation, but it completely changes the notion of a min cut set. The min cuts at each node are formed from each of its direct children, viewing each as a component even for those that are gates. But alas, life is usually not that nice. The reason is that many nodes in most fault tree systems have multiple parents. This means that failure can move up the tree in more than one path. This changes the nature of the tree from a hierarchy to a network, and consequently greatly complicates the min cut set generation as well as the analysis.

When failure can propagate up the tree in various pathways, the cuts themselves become more complex. The same components now are now found in a variety of branches. One can no longer rely on the cuts feeding a particular node, since they may overlap with cuts in a wholly different branch. The may also form cuts in one branch that are supersets of cuts in another and should be removed higher up in the tree.

Figure 3 shows such an example in the rear wheel assembly.

Figure 3
Rear Wheel Branch: Network



The rear axle in the "steering" branch is affects both rear wheel assemblies. Consequently the same basic event is shared between these two branches.

In this case, neither the LEFT or RIGHT branches may be used as independent branches. Cuts developed at either of these two nodes must be carried upward to the next higher node. But at this point, the min cut sets can be resolved, and the node BACK can be treated as an independent branch for processing further up the tree.

So in practice, if a fault tree algorithm is able to identify all independent branches, it can gain significant computation efficiency, both in CPU time and memory used.

IMPORTANCE AND SENSITIVITY

As a final topic we will entertain establishing the influence a single component has on the reliability of some system. Starting with the cuts and associated failure probabilities of table 3, one could and should ask how much change would occur to the total reliability if the failure rate of some component changed. Or more drastically, what impact would there be if a component become perfectly reliable or completely unreliable?

Multiple approaches exist addressing this. One approach is to view the failure probability calculation as a reliability formula (actually unreliability) which varies as a function of the components' failure rates. If one changed the failure rate by some amount, and compared this change to the resulting change in the failure probability, this comparison could be viewed as a slope. This slope is the change in failure probability per unit change in the failure rate. It can be constructed by taking the partial derivative of the reliability formula with respect to each failure rate, resulting in a vector of we might call ***importance measures***. The larger the measure, the bigger influence that component's reliability has on the mission reliability. Suppose we calculate this importance measure for each component. The components are then ranked in terms of their values is descending order. The components are then ranked. This ranking then provides a useful comparison of relative importance of the components in the system.

Table3
Min Cut Set for Tricycle Application

	Events	Phase 1		Mission	
No	TOP	1.018E-01	Rnk	1.018E-01	Rnk
1	XPNUT1	2.155E-01	1	2.155E-01	1
2	XPBLT2	2.155E-01	1	2.155E-01	1
3	XPBLT1	2.155E-01	1	2.155E-01	1
4	XPNUT2	2.155E-01	1	2.155E-01	1
5	XCLAMP	1.196E-01	2	1.196E-01	2
6	XFLNUT	1.083E-01	3	1.083E-01	3
7	XFRNUT	1.083E-01	3	1.083E-01	3
8	XSEAT1	1.083E-01	3	1.083E-01	3
9	XFDLM	1.083E-01	3	1.083E-01	3
10	XCBOLT	1.083E-01	4	1.083E-01	4
11	XRAXL	2.276E-02	5	2.276E-02	5
12	XHUB1	1.138E-02	6	1.138E-02	6
13	XCNUT	1.138E-02	6	1.138E-02	6
14	XHUB2	1.138E-02	7	1.138E-02	7
15	XRDLM	1.089E-02	8	1.089E-02	8
16	XLDLM	1.089E-02	8	1.089E-02	8
17	XFRAME	1.089E-02	8	1.089E-02	8
18	XFNUT2	2.392E-03	9	2.392E-03	9
19	XFNUT1	2.392E-03	9	2.392E-03	9
20	XFBLT2	2.392E-03	9	2.392E-03	9
21	XFBLT1	2.392E-03	9	2.392E-03	9
22	XRGRIP	1.196E-03	10	1.196E-03	10
23	XLGRIP	1.196E-03	10	1.196E-03	10
24	XFAXL	1.089E-03	11	1.089E-03	11

Importance measures & ranking allows the reliability engineer to focus on elements of the design which are sensitive to the overall system reliability. It provides the basis for adding redundancy to a design or for focusing efforts on particular component that yield the greatest payback in quality. This then begs an issue with respect to joint coordination between Quality and Engineering departments, since the reliability can be a common denominator with which Engineering can effectively guide fruitful areas of quality improvement.

OTHER FAULT TREE CONSIDERATIONS

There are a variety of other key issues that we must leave for another day. These issues are left not because of their relevance, but their complexity. Below is a list of some of methods that are used in practice, but not discussed in this paper.

- Up to this point, we have assumed that when a component fails, it stays failed. But in practice, some systems allow for replacing or repairing components that fails, so long as the system has not yet failed. ***Repairable systems*** are far more complex and are generally approximated or analyzed using Monte Carlo methods.
- The ***k choose n gate*** allows a fixed number of child nodes to fail before the parent gate is considered failed.
- Another gate called a ***sequential AND gate*** considers an AND gate to be failed only if the failure occurs in a particular order.
- The concept of a "mission" cane be generalized using the notion of a ***time phased mission***. In a phased fault tree, the failure rates can change at certain block points where situations may incur greater stress on certain components producing higher risk.
- Certain fault tree switches called ***houses*** can turn on and off particular branches as a function of the phase, allowing for tailoring the fault tree within the mission.
- Often certain gates fail not only as a function of the cutsets, but as a function of a random, non-time oriented event. Such evens are often called ***inhibit conditions***. These conditions are like houses, except they occur randomly. The rate of occurrence can change from phase to phase.

CONCLUSION

Fault Tree Analysis is one of many methods that can be used to evaluate and engineer reliability into a product or service. Reliability methods can also provide a basis for handshaking with Quality departments, enabling a focus on fruitful quality investigation or improvement. The methods surrounding Fault Tree Analysis are cause and effect in nature, with probabilistic models being the primary tool for quantifying the system reliability. Such tools are powerful, opening the door to superior and more cost effective systems leading to customer satisfaction and trust.

REFERENCES

- Fault Tree Handbook: NUREG-0493; U.S. Nuclear Regulatory Commission
- SAFTE (Simulation & Analytic Fault Tree Evaluator) The Boeing Company
- Merriam-Webster Online Dictionary
  http://www.m-w.com/cgi-bin/dictionary?book=Dictionary&va=Reliability&x=19&y=14
- Huffy 9" Disney Princess Low Rider
  http://www.mytoybox.com/